Professional and Scientific Staff Management, Inc.
Temporary Staff Fulfillment System
Deployment Proposal
2016-12-07

Rob Sanchez and Juan Carcamo
Knowledge Machines Analysis & Design

CIS 641, Fall 2016
Schymik

Contents

1. Introduction

The purpose of this document is to present a deployment proposal for a Temporary Staffing Fulfillment System (*TechStaff*) at Professional and Scientific Staff Management, Inc. of Toledo, OH. PSSM is a medium-sized, niche-market staffing agency operating in the Midwestern United States. It specializes in high-skilled technical and scientific temporary staffing placements and has experienced an unexpected surge in demand for suitable temporary staffing candidates from companies throughout the region in the last three years.

PSSM's current placement and arrangements system, such as it is, hasn't been able to keep up with the agency's growth and suffers from significant performance and usability problems. As a result, leadership has decided to replace it. They reached out to our firm, *Knowledge Machines Analysis & Design*, for a re-implementation of their solution.

PSSM's core competency is negotiating contracts and maintaining relationships between their counter-parties. They employ a minimal IT team with no plans to hire new staff. Their current staffing system consists of a set of related spreadsheets stored on the company's shared network drive. This arrangement does not support a high level of concurrency and suffers from a high rate of errors. It also doesn't protect client or candidate staff personal data to the satisfaction of the PSSM legal department.

PSSM are looking for a hosted, cloud-based web application to access from their offices in downtown Toledo, with physical data management to be handled offsite by a third party. Basic web application configuration access should be available to PSSM IT staff, but code development and maintenance is to be managed by KMAD through a standard support contract.

2. Project Description

2.1 Requirements Overview

The main business requirement the *TechStaff* system will meet is that of Staffing Request Fulfillment. To describe this process briefly, PSSM's clients phone or email their demands for temporary staffing placements, which PSSM attempts to fill by locating potential candidates. A set of criteria are given by the client and PSSM matches them in the availability pool.

The two main benefits to PSSM for implementing *TechStaff* will be concurrency and data integrity. The current spreadsheet-based solution does a poor job of handling these requirements, and as PSSM continues to grow, a more effective solution will be needed. Whereas PSSM staff now have to coordinate among themselves for access to the staffing data, KMAD's TechStaff solution will allow multiple members of staff to interact with the system at the same time, as well as store the data itself more securely. PSSM's current file-based network storage approach poses significant risks to the business, and moving the data to more reliable database technology is a logical approach.

As a result, PSSM's existing spreadsheet data will need to be migrated into the *TechStaff* database. This process is relatively straight-forward: currently three spreadsheets are used: Contracts, Employees, and Requests. These more or less already fit the data model required. Each spreadsheet

will represent a table in the *TechStaff* database, with a web application on top to manage the data and help PSSM execute their core business processes.

2.2 Business Processes

After a number of discussions between PSSM and KMAD, it was determined that Staffing Request Fulfillment consists of three main business processes:

1) Opening a staffing request,
2) Placing a candidate, and
3) Filling the staffing request.

The people that interact with these business processes are (in SAD parlance, "actors") PSSM Clients, the PSSM Contract Manager, the PSSM Placement and Arrangements departments, and Temporary Employees in the candidate pool.

Before design and implementation can begin for any new information system, a significant amount of planning and analysis time is required first. In the sections that follow, we present a series of planning and analysis step that will become crucial to our effort for preparing the system design.

For each business process, we being by functionally modeling a use-case and associated use-case description. These describe the interactions between actors and the system to achieve the goal of executing the required business processes. These are detailed in sections 4.1.1 and 4.1.3 in the Functional Analysis section below. In addition, we've detailed the overall process flow for the fulfillment process in an Activity Diagram (section 4.1.2). We use these functional analysis design tools ("artifacts") to build the structural models of the system (section 4.2).

3. Effort Estimation

In order to estimate the level of effort required to deliver the solution, we've assembled a use case points analysis to estimate the level of effort in person-hours. A use case points analysis operates on 4 main levels: 1) Actors, 2) Use Cases, 3) Technical Complexity, and 4) Environmental Factors. We use an industry-standard (Karner) use case points analysis model to determine the estimate.

3.1 Actors

Actors are counted and weighted according to their complexity. This results in an Unadjusted Actor Weight Total (UAW). The *TechStaff* application includes 1 average actor (the database) and 4 complex actor types (endusers).

Unadjusted Actor Weight Total (UAW): [14]

3.2 Use Cases

Use cases are also counted and weighted according to their complexity. This results in an Unadjusted Use Case Weight Total (UUCW). The *TechStaff* application includes 3 average use cases (4-7 transactions each).

Unadjusted Use Case Weight Total (UUCW): [30]

3.3 Unadjusted Use-Case Points

The UAW and UUCW values are added together to yield UUCP, or Unadjusted Use-Case Points. This is used in the following Technical Complexity and Environmental Factors sections.

Unadjusted Use-Case Points (UUCP) = UAW + UUCW: 14 + 30 = [44]

3.4 Technical Complexity

The technical complexity of the project is estimated by considering and weighting a collection of technical factors, such as response time expectations, ease of use, portability, and concurrency. In particular, PSSM's solution requires more concurrency than currently available, because they have multiple staff members from Contracts, Placements and Arrangements requiring use of the system at the same time. This part of the analysis yields a Technical Factor Value (TFactor). This in turn is converted into a Technical Complexity Factor measurement according to an established formula in our Karner model.

Technical Factor Value (TFactor): [34.5]

Technical Complexity Factor (TCF) = 0.6 + (0.01 * TFactor): [0.945] = 0.6 + (0.01 * 34.5)

3.5 Environmental Factors

Other environmental factors are considered to determine the effort estimate of the project. Criteria such as familiarity with the system development process, the team's object-oriented experience, stability of requirements, and whether part-time staff are to be used in the project all play a role. After quantifying their impact, this part of the analysis yields an Environmental Factor Value (EFactor). This in turn is converted into an Environmental Factor measurement according to an established formula in our Karner model.

Environmental Factor Value (EFactor): [25]

Environmental Factor (EF) = 1.4 + (-0.03 * EFactor): [0.65] = 1.4 + (-0.03 * 25.5)

3.6 UCP and PHM

With these data we can now calculate Adjusted Use Case Points (UCP) as well as the Person-Hours Multiplier (PHM). UCP adjusts our raw points estimate from step 3.3 for Technical and Environmental factors, and the PHM is used to estimate the number of person-hours required to build the system. Depending on the environmental factors score, this value could be 20 or 28. For PSSM *TechStaff*, the value is 20.

Adjusted Use Case Points (UCP) = UUCP * TCF * ECF: [27.027] = 44 * 0.945 * 0.65

Effort in person-hours = UCP * PHM: [540.54] = 20 * 27.027

3.7 Final Effort Estimate

After this set of calculations, using our Karner use-case points analysis model, we can estimate that it will take just over 540 hours to build the *TechStaff* solution for PSSM.

Effort in person-hours: [540. 54 hours]

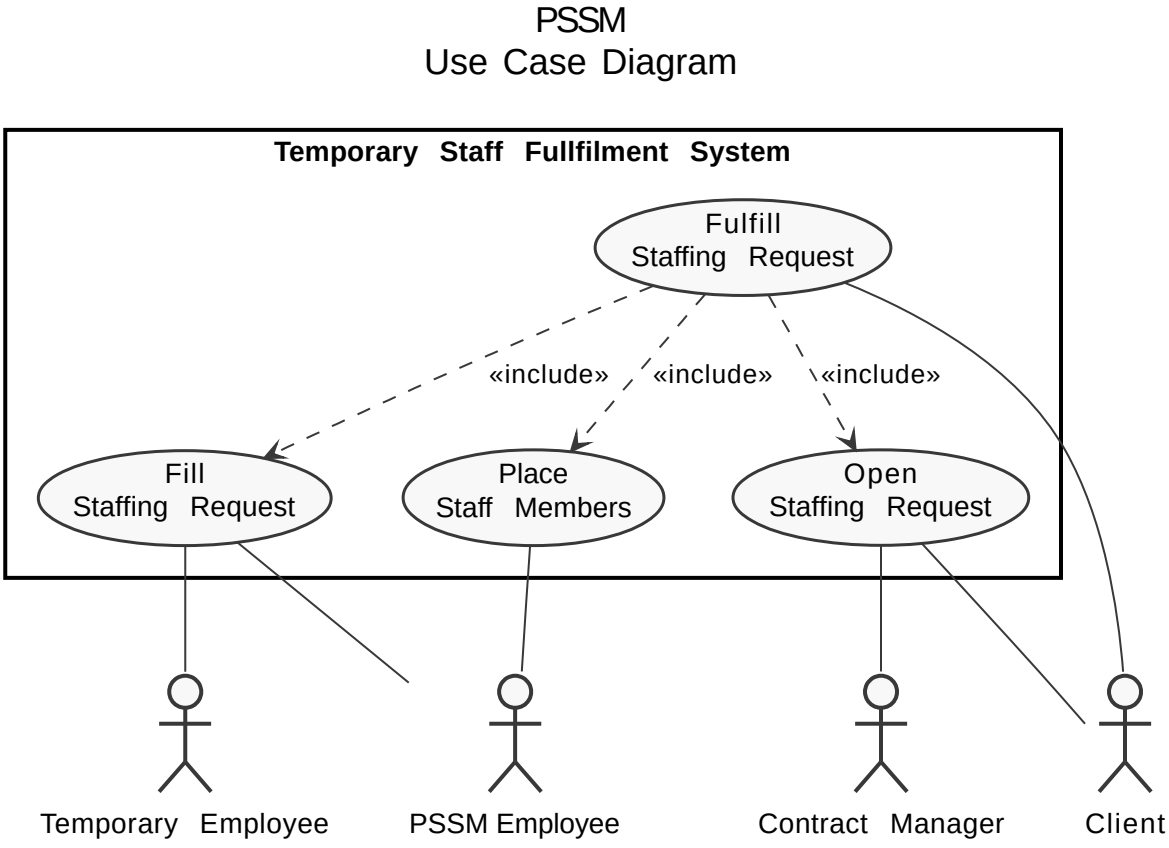
4. Analysis

4.1 Functional Models

Functional analysis models are used to document the functions the software is intended to perform, from the perspective of the business user. Artifacts such as use case diagrams, activity diagrams, and use-case descriptions are used to communicate with endusers, and model their expectations. After a number of discussions with representatives from PSSM's business teams, we were able to model their expectations as shown below.

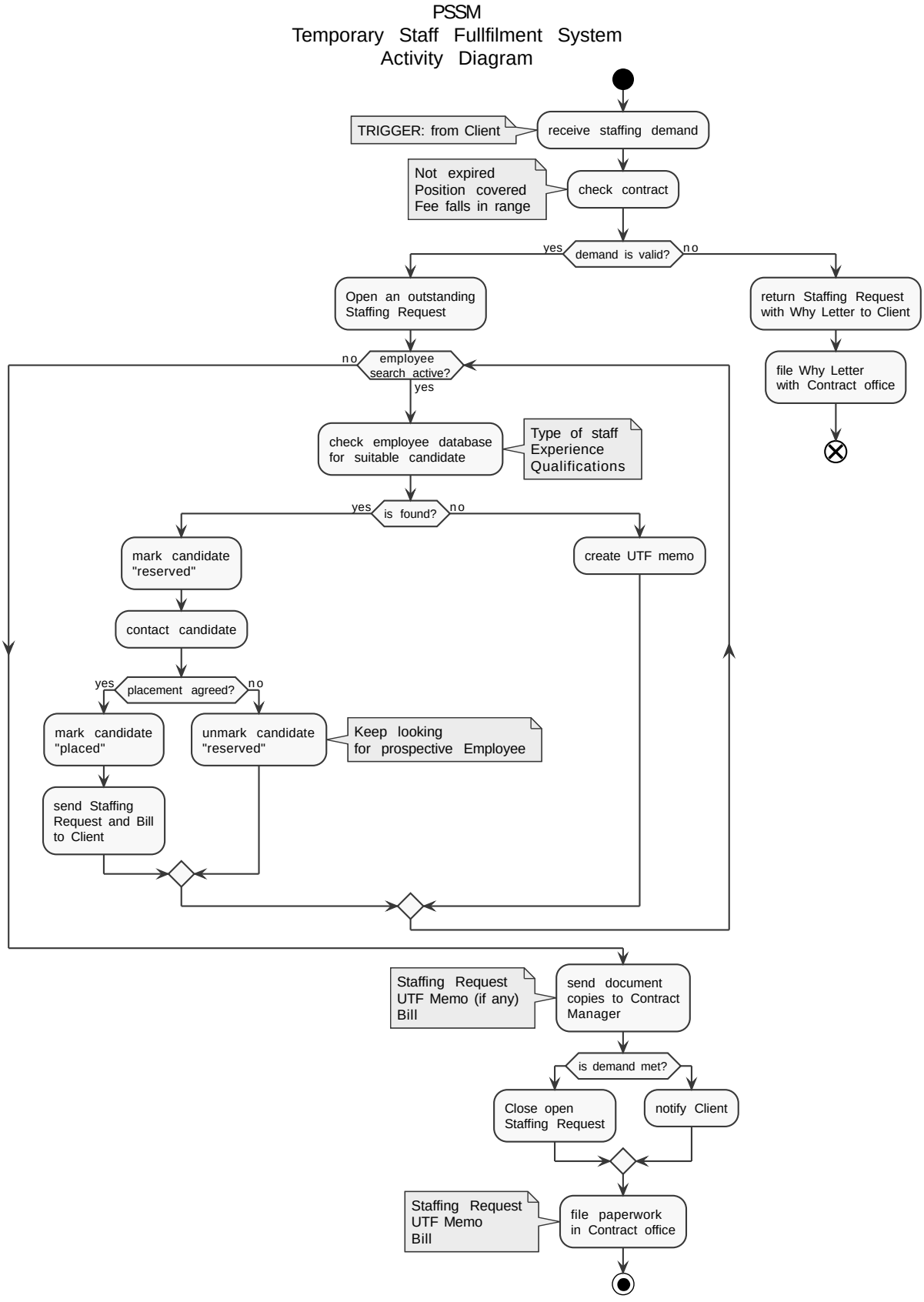
4.1.1 Use Case Diagram

The following Use Case Diagram depicts the main business processes performed required by the *TechStaff* system. At a high level, we the use case diagram helps us model the overall operation of the system, the actors involved, and any dependencies on outside entities or systems. In this case, the overview use case "Fulfill Staffing Request" *includes* three detailed use cases, which correspond to the main business processes we mentioned above in section 2.2.



4.1.2 Activity Diagram

The Activity Diagram demonstrates the flow of activities during Staffing Request Fulfillment.



4.1.3 Use Case Descriptions

The following Use Case Descriptions functionally document the business processes depicted in the Use Case Diagram and Activity Diagram presented in sections 4.1.1 and 4.1.2. All of these functional artifacts are used to build the structural models for the Staffing System.

Use Case Name:	Open Staffing Request (detailed)
Scenario:	Create a new outstanding Staffing Request.
Triggering event:	Client contacts PSSM via phone or email and demands a new temporary staff arrangement.
Brief description:	PSSM receives a new demand for temporary staffing from the Client. PSSM's Contract Manager reviews the contract terms. If the new demand is not valid, the demand is returned to the Client with a Why Letter explaining. Otherwise, a new Staffing Request is opened in the Contract database and Placement dept. is notified.
Actors:	Client, Contract Manager
Related use cases:	Place Staff Members, Fulfill Staffing Request
Stakeholders:	Contract Manager, Temporary Employee, Placement department
Pre-conditions:	A valid demand.
Post-conditions:	A new outstanding Staffing Request is created and the Placement department is notified.

Flow of activities:		
	Actor	System
	1. Client demands a new temporary staff arrangement 2. Contract manager reviews the client contract 3. Contract manager opens a new Staffing Request	2-1. Referencing the contract number on the new demand, the System pulls the Client contract details from the database. 3-1. System creates a new Staffing Request, marking it outstanding 3-2. System notifies Placement department of outstanding Staffing Request.

Exception conditions:	2-1. Demand is invalid, send Why Letter to Client explaining. File letter locally.

Use Case Name:	Place Staff Members (detailed)
Scenario:	Find suitable temporary staff to fill a client demand.
Triggering event:	Contract Manager opens a new outstanding Staffing Request, and the System notifies the Placement department.
Brief description:	Placement department checks position type, experience, and qualifications specified on the Staffing Request in the staff database.
Actors:	Placement department: PSSM Employee
Related use cases:	Open Staffing Request, Fill Staffing Request, Fulfill Staffing Request
Stakeholders:	Placement department, Contract Manager, Arrangements department, Temporary Employee
Pre-conditions:	A new outstanding Staffing Request must exist. A qualified Temporary Employee must be available.
Post-conditions:	A qualified Temporary Employee is marked as "reserved" and the Arrangements department is notified.

Flow of activities:		
	Actor	System
	1. A PSSM Employee from Placement checks the position type, experience, and qualifications specified in the Staffing Request against the database of available Temporary Employees. 2. The Temporary Employee is marked as "reserved"	1-1. System pulls available Temporary Employees matching the Staffing Request. 2-1. The System forwards the Staffing Request to the Arrangements department and notifies them that a possible match was found.

Exception conditions:	2-1. A qualified Temporary Employee is not available and an "unable-to-fill" Memo explaining is attached to the Staffing Request. Staffing Request is forwarded to the Arrangements dept.

Use Case Name:	Fill Staffing Request (detailed)
Scenario:	Arrange a Temporary Employee assignment for a Client.
Triggering event:	A possible qualified and available staff candidate is marked as "reserved" in the System, and Arrangements department is notified.
Brief description:	A potential temporary staff candidate is contacted, and after settling details and negotiating terms, the Temporary Employee is placed with the client and the Staffing Request is Closed.
Actors:	Arrangements department: PSSM Employee, Temporary Employee
Related use cases:	Place Staff Members, Fulfill Staffing Request
Stakeholders:	Client, Temporary Employee, Arrangements department
Pre-conditions:	A possible qualified and available temporary staff candidate has been marked as "reserved" in the System.
Post-conditions:	A Temporary Employee is marked as "placed" in the System. The open Staffing Request is closed. A copy of the original Staffing Request and a Bill is sent to the Client. The Staffing Request, any "unable-to-fill" memos and a copy of the Bill are sent to the Contract Manager.

Flow of activities:		
	Actor	System
	1. A PSSM Arrangements Employee contacts the temporary staff candidate. 2. The PSSM Employee settles any details and negotiates terms with the candidate. 3. The PSSM Employee places the Temporary Employee with the Client. 4. The PSSM Employee closes the open Staffing Request.	3-1. The System marks the Temporary Employee as "placed". 4-1. The System closes the open Staffing Request. 4-2. Copy of the Staffing Request and a Bill are sent to the client. 4-3. The Staffing Request, a copy of the Bill, and any "unable-to-fill" Memo is sent to the Contract Manager.

Exception conditions:	3-1. Staffing Request not filled, client is notified. Staffing Request, Bill, and "unable-to-fill" Memo are filed in the contract office.

Use Case Name:	Fulfill Staffing Request (overview)
Scenario:	Open and fill a Staffing Request in order to place a Temporary Employee with a Client to meet a demand.
Triggering event:	A Client contacts PSSM to demand a new temporary staff arrangement.
Brief description:	The PSSM Contract Manager receives the Client demand and reviews the contract details. The Placement department finds a suitable staff member to meet the Client demand. Arrangements department negotiates the arrangement and places the employee with the Client.
Actors:	Client
Related use cases:	Open Staffing Request, Place Staff Members, Fill Staffing Request
Stakeholders:	Client
Pre-conditions:	Client and PSSM have an active contract in place.
Post-conditions:	A Temporary Employee is placed with the Client.

Flow of activities:		
	Actor	System
	1. Contract Manager opens a Staffing Request. 2. The Placement department identifies a suitable staffing candidate. 3. The Arrangements department negotiates with and places the Temporary Employee with the Client.	1. Placement department is notified. 2. Arrangements department is notified. 3. The Client is billed.

Exception conditions:	2-1. The Staffing Request cannot be fulfilled, the client is notified. The SR, Bill, and “unable-to-fill” Memo are filed in the Contract office.

4.2 Structural Models

4.2.1 CRC Cards

CRC cards, or class-responsibility-collaboration cards, are used to document the responsibilities and collaborations of a class. In object-oriented analysis and design, classes are used to represent and model real-world objects in software to get work done. CRC cards include a list of class components, methods associated, and relationships with other classes, if any. The PSSM *TechStaff* solution includes three classes: Contract, TempEmployee, and StaffingRequest. Their CRC cards are shown below.

Front:

Class Name: Contract	ID: 1	Type: Concrete
Description: A concrete class representing contracts negotiated between PSSM and its Clients		Associated Use Cases: 3
Responsibilities		Collaborators
{ CRUD }		
Add to Contracts Table		{ database }
Assign to Staffing Request		StaffingRequest
Assign to Employee		TempEmployee
Add to Employee (previous contracts)		TempEmployee

Back:

Attributes:
Contract ID (int)
Expiration Date (date)
Terms { free text field }
Client Name (string)
Client Address (string)
Client Phone (string)
Client Email (string)
Client Contact Person (string)
Placed Employees (string array)[ ]<EmployeeID>

Relationships:	
Generalization (a-kind-of):	
Aggregation (has-parts):	
Other Associations:	TempEmployee, StaffingReqeust

Front:

Class Name: TempEmployee	ID: 2	Type: Concrete
Description: A concrete class representing employee candidates to be placed for temporary employment with Clients of PSSM		Associated Use Cases: 3

Responsibilities	Collaborators
{ CRUD }	
Add to Employees Table	{ database }
Add to Contract (placed employees)	Contract
Reserve	StaffingRequest
Place	StaffingRequest

Back:

Attributes:

- EmployeeID (int)
- Employee Status (string)[available, reserved, placed]
- Employee Name (string)
- Employee Address (string)
- Employee Phone (string)
- Employee Email (string)
- Years of Experience (int)
- Degree (string)[bachelor's, master's, doctorate]
- Qualifications { free text field }
- Expected Pay (float)
- Current Contract (int)<ContractID>
- Previous Contracts (int array)[ ]<ContractID>

Relationships:	
	Generalization (a-kind-of):
	Aggregation (has-parts):
	Other Associations: Contract, Staffing Request

Front:

Class Name: StaffingRequest	ID: 3	Type: Concrete
Description: A concrete class representing staffing requests used to manage placements and arrangements between employee candidates and positions with PSSM clients		Associated Use Cases:

Responsibilities	Collaborators
{ CRUD }	
Add to Requests Table	{ database }
Assign to Contract	Contract
Reserve Candidate	TempEmployee
Place Employee	TempEmployee
Close	{ database }

Back:

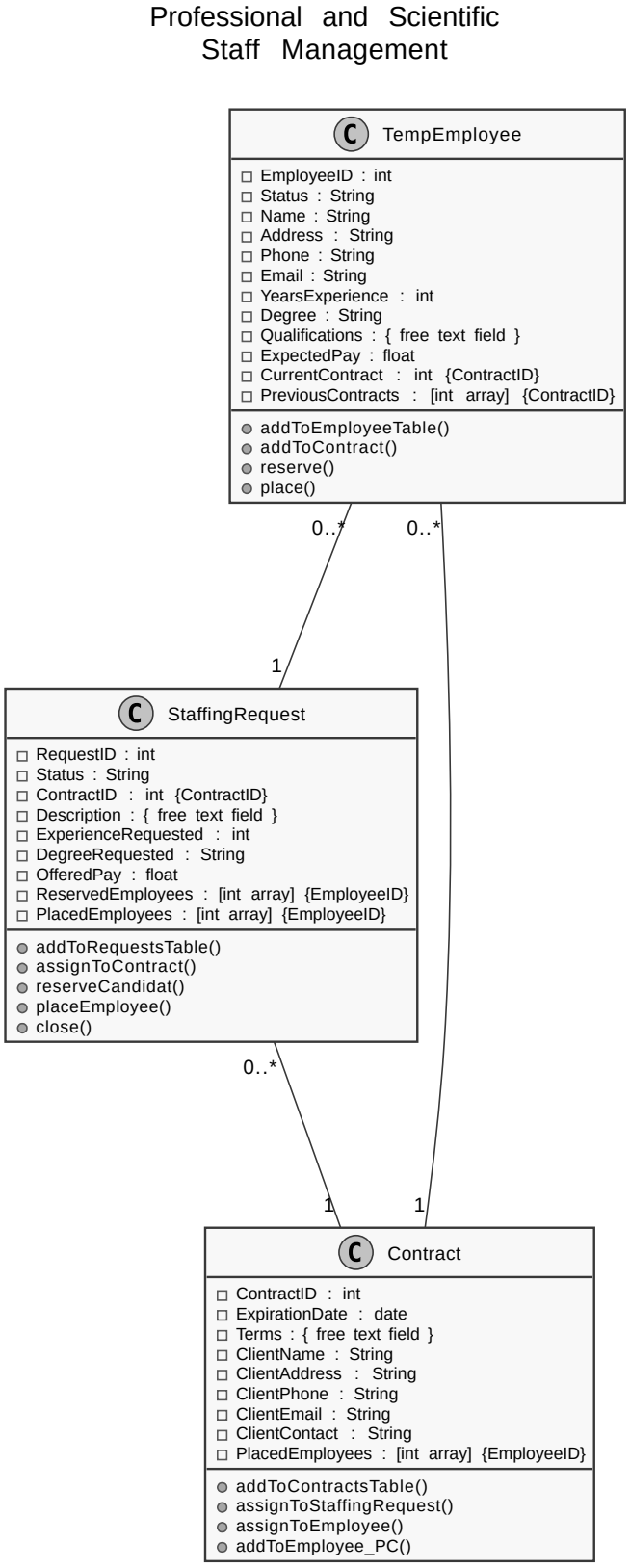
Attributes:

- RequestID (int)
- Request Status (string)[open, closed]
- Contract ID (int)<ContractID>
- Description { free text field }
- Experience Requested (int)
- Degree Requested (string)[bachelor's, master's, doctorate]
- Offered Pay (float)
- Reserved Employees [int array]<EmployeeID>
- Placed Employees [int array]<EmployeeID>

Relationships:		
	Generalization (a-kind-of):	
	Aggregation (has-parts):	
	Other Associations:	Contract, TempEmployee

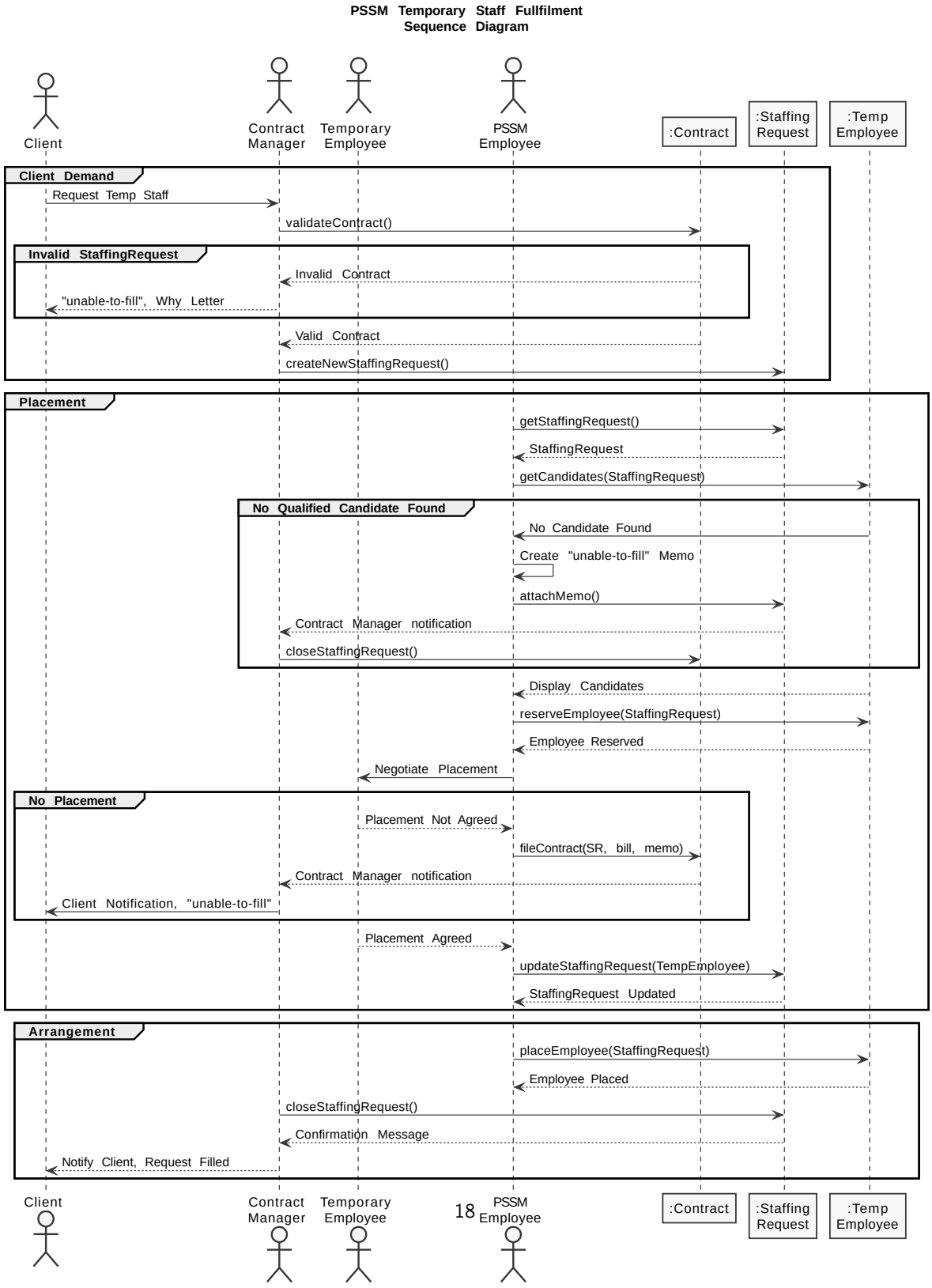
4.2.2 Class Diagram

The class diagram is a static structural model that shows the relationships between classes that remain constant in the system over time. Class behaviors and states are documented as shown below. The classes documented in the CRC cards above are depicted below.



4.2.3 Sequence Diagrams

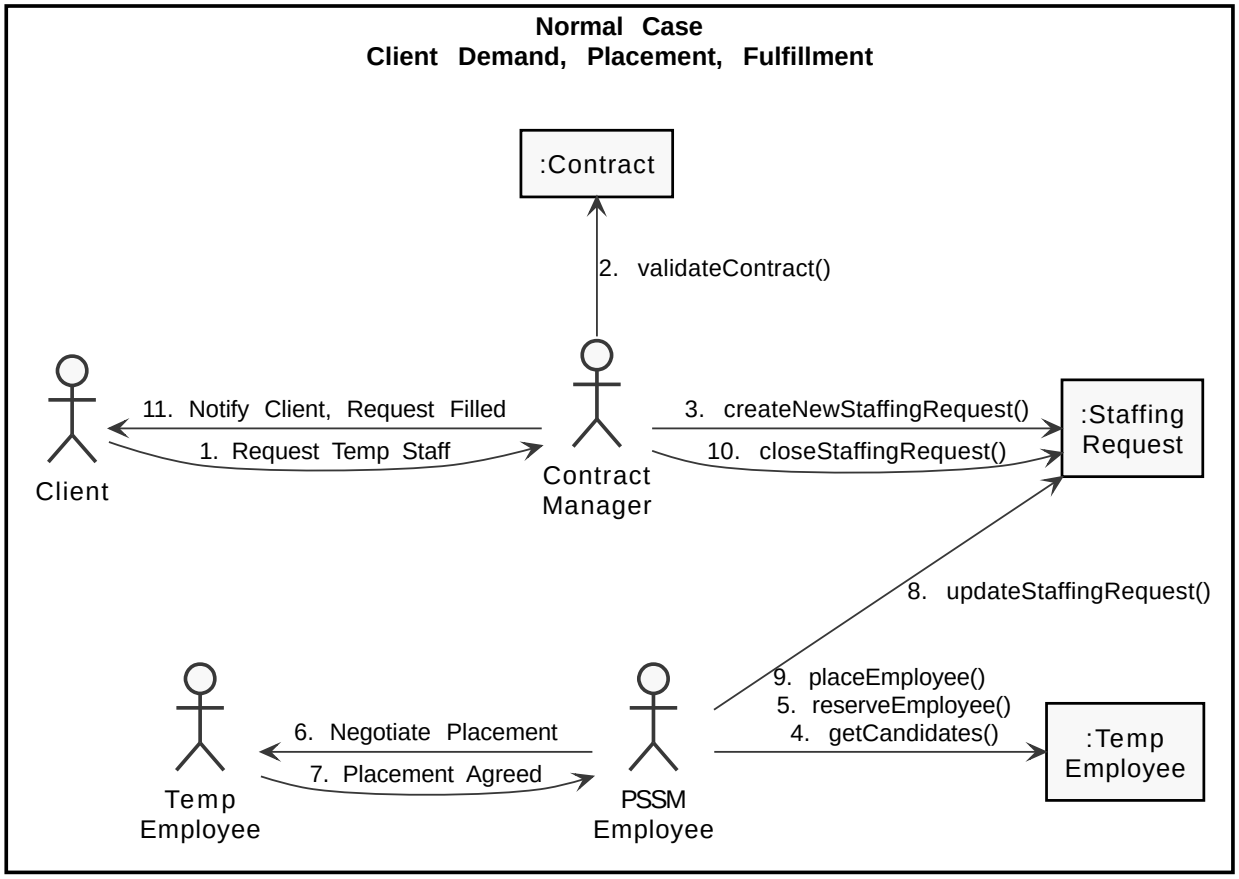
A sequence diagram is a dynamic behavioral model showing the sequence of messages passed between objects in an interaction. In general, sequence diagrams are time-based so the order of operations is depicted.



4.2.4 Communication Diagrams

A communication diagram is similar to a sequence diagram in that it's a dynamic behavior model of the interactions between objects in the system, but the focus is on the messages themselves rather than the time or order they occur in. Communication diagrams can show how dependent classes are on each other.

PSSM Temp Staff Fullfilment
Communication Diagram



4.2.5 Behavioral State Machine

< insert behavioral state machine here >

4.2.6 CRUD Matrix

< insert CRUD matrix here >

4.2.7 Verification & Validation of Functional, Structural, and Behavioral Models

In general, the functional, structural, and behavioral models must be verified and validated before design begins. This is to make the design effort go more smoothly, and reduce the amount of costly re-work necessary to complete the solution. The better the analysis is early, generally the fewer mistakes are discovered later on in the project. Fixing problems early is much cheaper than fixing them later. Here we present a verification and validation checklist for the functional, structural, and behavioral models.

Validation &Verification Checklist:

1. Functional Models:
   - [x] At least one normal use case description event for each activity in activity diagrams
   - [x] All activity diagram object nodes must be mentioned as events in use case descriptions.
   - [x] Use case description sequential order should match that of the activity diagram.
   - [x] Only one use case description per use case.
   - [x] All actors in the use case description must be portrayed in the use case diagram.
   - [x] All relationships in the use case description must be portrayed in the use case diagram.

2. Structural Models:
   - [x] CRC cards → classes on class diagrams
   - [x] CRC card responsibilities → class diagram operations
   - [x] CRC card collaborators → CRC card relationships & class diagram association classes
   - [x] CRC card attributes → class diagram attributes
   - [x] CRC relationships → appropriate notation
   - [x] Association classes should only reflect unique characteristics about the intersection of connecting classes.

3. Behavioral Models:
   - [x] All actors and objects in sequence diagrams included in communication diagrams.
   - [x] Sequence diagram messages → communication diagram associations
   - [x] Sequence diagram messages → communication diagram messages
   - [x] Sequence diagram guard conditions → communication diagram guard conditions
   - [x] Communication diagram sequence numbers reflect sequence diagram order
   - [x] Behavior state machine transitions → sequence and communication diagram messages
   - [x] Behavior state machine transitions → CRUDE matrix classification
   - [x] CRUDE matrix entries → messages between actors and objects, behavior state machine transitions

5. Design

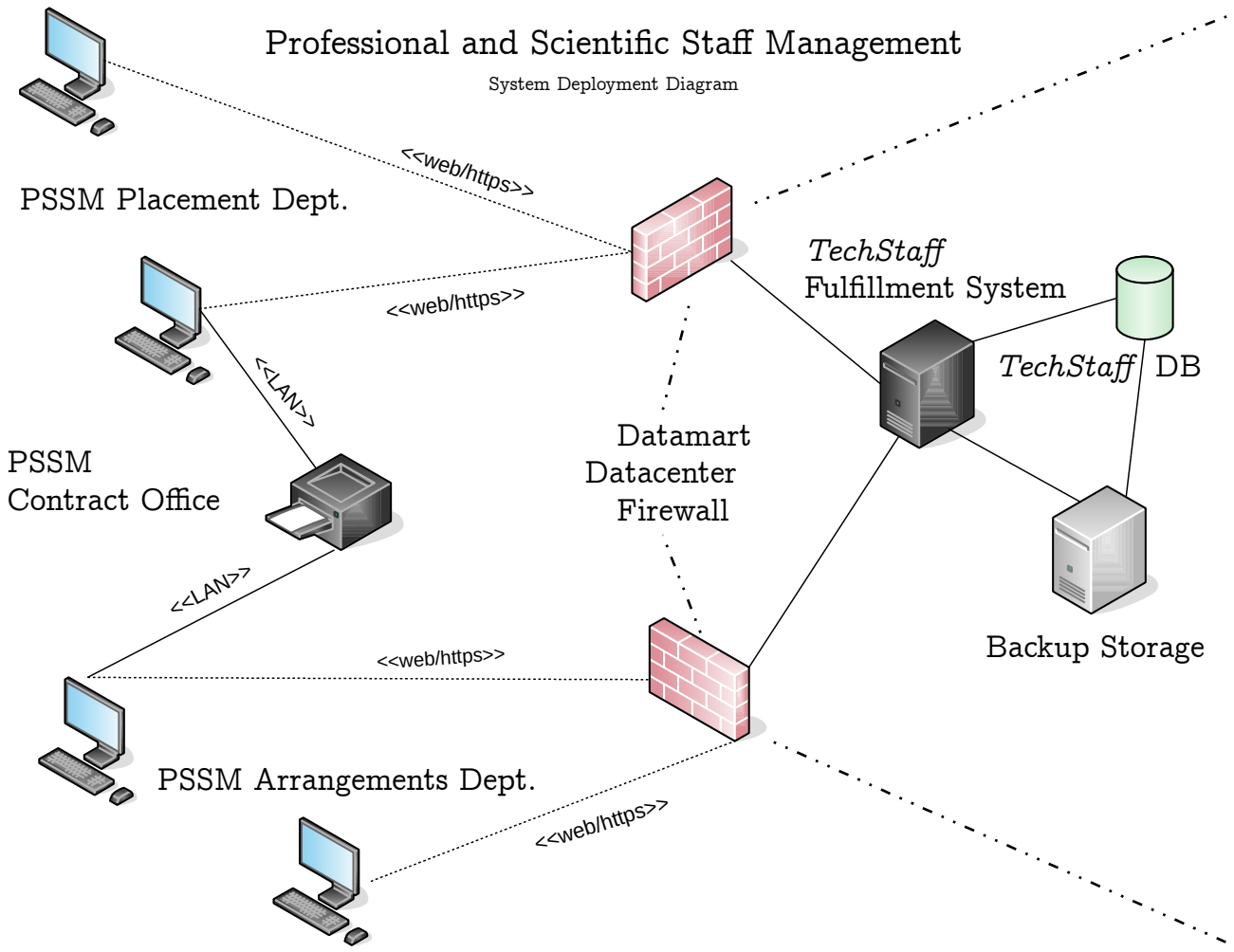
   The system will be named *TechStaff*.

6. User Interface

7. Physical Architecture

7.1 System Deployment Diagram

The deployment diagram below presents the physical architecture of the *TechStaff* application with respect to PSSM. The web application is to be hosted at Datamart, an industry-leading third-party datacenter provider. Datamart will be in charge of physical data management and security, as well as backups and routine system (OS, database & web server) patching. PSSM staff will access the *TechStaff* application via encrypted HTTPS, over the web, but will print their reports locally. Web connections to the *TechStaff* application will traverse the Datamart datacenter a firewall to access the application. PSSM users don't require and won't have access to the database directly, nor to Datamart's backup appliance.



7.2 Non-Functional Requirements

PSSM have specified a number of non-functional requirements related to the physical architecture design, which we describe here. Non-functional requirements are requirements outside the scope of the business process requirements specified in the application software, and include performance, operational, and security considerations.

7.2.1 Operational

- *TechStaff* should be usable on multiple web browsers.
  - Due to security or performance issues, users are sometimes forced to temporarily or permanently switch to a new web browser. The application should follow industry standards and work adequately on compliant browsers such as Firefox, IE, Chrome, or Safari.

- *TechStaff* should write to a database, not flat files.
  - One of PSSM's major headaches in the past few years has been the shared network storage nature of their current spreadsheet file system for managing contracts, employees, and staffing requests. They currently have no support for concurrency, so staff must coordinate among themselves to make updates to the shared files.

7.2.2 Performance

- *TechStaff* should perform adequately well for 10 concurrent users.
  - While PSSM's performance expectations do not approach those of NASA, the military, or the NASDAQ stock exchange, at least 10 concurrent users should be supported at any given time by the application. This way the Contracts, Placement, and Arrangements team can all work on the system at the same time and improve overall business efficiency.

- *TechStaff* scheduled maintenance should occur on weekends or after hours.
  - Datamart, the third-party datacenter provider, has to perform system software patching at regular intervals to protect the integrity of all systems in its datacenter. The *TechStaff* system should only be down for maintenance after hours (7pm to 7am Eastern time) or at the weekend, when PSSM use is low. PSSM should be notified of the downtime patching calendar in advance.

7.2.3 Security

- *TechStaff* should be available via encrypted HTTPS over the web.
  - PSSM require that the system accommodate employees that sometimes work from home or remotely. In addition, they'd like to be ready for a future release where clients and/or employee candidates have direct access to the portal to approve placements.

7.2.4 Cultural & Political

- TechStaff should protect the privacy of PSSM clients and employee candidates.
  - PSSM's business relies on its discretion. It would be difficult for PSSM to remain in business if they made client and candidate information available to just anyone. PSSM's expertise revolves around maintaining good relationships between its counterparties and managing expectations, especially in the case of disputes. If private data were to get out, they could be sued or lose business. Datamart and KMAD will be responsible for working together to secure confidential data.

8. Testing

A System Test Plan will allow PSSM to gain confidence that KMAD understood PSSM's needs and requirements, and successfully implemented a useful solution to meet them. We propose a Testing Plan with intertwined and interrelated phases: Requirements Testing, Documentation Testing, Alpha Testing and Beta Testing. We provide a set of test cases below and make use of them during all four phases.

8.1 Requirements Testing

Requirements testing determines whether or not the solution meets the original business requirements. These will be performed by KMAD, using the test cases presented in section 8.5 below. This testing phase will occur in two major parts: 1) after each new system class is implemented, and 2) after the code for each major business process is committed. For example, after the Contract class is finished, test case #2 will be executed. Subsequently, after class TempEmployee is complete, test case #3 will be executed, etc. Similarly, after the code for reserving a candidate for a staffing request is committed, test case #5 will be executed.

8.2 Documentation Testing

Documentation testing determines whether or not the solution matches the documentation delivered by KMAD's documentation team. This documentation should be used by PSSM during hands-on training to familiarize staff with the use of the *TechStaff* application. This

documentation will be delivered separately, but KMAD will test that the documentation matches the test cases below (8.5) immediately before Alpha testing begins, described below.

8.3 Alpha Testing

Alpha testing determines whether or not the solution meets the customer requirements by using test data. This testing is performed by PSSM staff, with the guidance of KMAD. Again, after each class and after each business process are implemented, the appropriate test case will be provided to PSSM staff for execution. Following the Agile/Scrum methodology, this puts the solution into the hands of the customer as quickly as possible, so that KMAD might gain valuable feedback and make adjustments early on, if necessary.

8.4 Beta Testing

Beta testing is the same as alpha testing, except that it's performed with real data in a so-called "staging" system. This means that PSSM's existing spreadsheet system will be unaffected, and PSSM operations will be unchanged until the new *TechStaff* solution is accepted and approved for go-live. This testing will again be performed by PSSM staff, under KMAD supervision. After beta testing in the staging system is user-accepted by PSSM, *TechStaff* can go-live and designate the newly-created staging system as PSSM's production system moving forward.

8.5 Test Cases

As discussed, the following test cases will be used in all four testing phases outlined above.

Test case #: 1	*Test Case Name:* Login, logout; Reset password
System: TechStaff	*Subsystem:* Security
Designed by: Larry David	*Design Date:* 2016-11-15
Executed by: Cosmo Kramer	*Execution Date:* 2016-11-30
Short Description: Test the option code field during sales Invoice data entry.	

Pre-conditions: The user has opened a web browser.

Step	*Action*	*Expected System Response*	*Pass/ Fail*	*Comment*
1	Enter the PSSM *TechStaff* URL in the browser.	The System displays the PSSM *TechStaff* web page. The user is		

		prompted for a username and password.		
2	Enter the assigned username and password. Click the "Login" button.	The System displays the *TechStaff* home screen.		
3	At the home screen, click Logout.	The System displays the PSSM *TechStaff* web page. The user is prompted for a username and password.		
4	Below the login fields, click the "Forgot password?" link.	The System displays an input field labeled, "Please enter your email address to reset your password"		
5	Enter the new email address. Click "Reset my password".	The System sends an email to the user.		
6	The user checks their email and locates the email sent by *TechStaff*. The user clicks on the reset password URL included in the email.	The System opens a new browser window, connects to the *TechStaff* application, and displays two input fields, for entering and confirming the new password.		
7	Enter and confirm the new password. Click Save.	A message is displayed, "Your password has been successfully reset. You may now log into the system.		
8	*Check post-condition 1.*			

Post-conditions:

1. The user is able to log into PSSM *TechStaff* with the newly-reset password.

Test case #: 2 *Test Case Name:* CRUD Contract

System: TechStaff *Subsystem:* Contracts

Designed by: Wade Wilson *Design Date:* 2016-11-15

Executed by: Bruce Wayne *Execution Date:* 2016-11-30

Short Description: Test the Creation, Reading, Updating, and Deleting of a client Contract

Pre-conditions:

The Customer has established a contract with a client.

The Customer has logged into the *TechStaff* web application.

Step	*Action*	*Expected System Response*	*Pass/*	*Comment*

			Fail	
1	Test Create: Click the "Create Contract" button	The system presents a list of input fields. Fields surrounded by a red box are required.		
2	Fill in required Contract fields: ID, Expiration Date, Company Name, Address, Phone, Terms	The system enables the Save button at the bottom of the input fields		
3	Click Save.	The system presents a pop-up message "Contract #<ContractID>" has been saved. The list of existing contracts is displayed.		
4	*Check post-condition 1.*			
5	Test Read: Click the "Find Contract" button	The system presents a list of Contract search input fields: ID, Expiration Date, Name, Phone		
6	Type the desired Contract ID into the "ID" field. Click "Search"	The system presents a list of matching contracts.		
7	Select the desired Contract and click the "Open" button	The system displays the Contract details. The fields are read-only.		
8	*Repeat Test Read (5) for Expiration Date, Name, and Phone.*			
9	Test Update: Open a Contract as described in Test Read (5) above.	The system displays the Contract details. The fields are read-only.		
10	Click the "Edit" button	The Contract fields become editable, except for the Contract ID field, which can't be changed.		
11	Enter a change for the Phone field. Click Save.	The system presents a pop-up message "Contract #<ContractID>" has been saved.		
12	*Check post-condition 2.*			
13	*Repeat Test Update (9) for Expiration Date, Name, and Phone.*			
14	Test Delete: Open a Contract as described in Test Read (5) above.	The system displays the Contract details. The fields are read-only.		
15	Click the "Edit" button	The Contract fields become editable, except for the Contract ID field, which can't be changed.		
16	Click the "Delete Contract" button	The system displays a pop-up message "Are you sure you want to delete Contract #<ContractID>? You cannot		

		undo this action." Two buttons are presented below: "Cancel" and "Delete"		
17	Click Delete.	The system displays a message, "Contract #<ContractID> has been deleted."		
18	*Check post-condition 3.*			

Post-conditions:

1. The new Contract is saved in the database with the entered data.
2. The updated Contract is saved in the database with the updated data.
3. The Contract has been removed from the database.

Test case #: 3	*Test Case Name:* CRUD TempEmployee
System: TechStaff	*Subsystem:* Employee
Designed by: Barry Allen	*Design Date:* 2016-11-15
Executed by: Doctor Strangelove	*Execution Date:* 2016-11-30

Short Description: Test the Creation, Reading, Updating, and Deleting of a client Contract

Pre-conditions:

The Customer has established a new relationship with a staff candidate.
The Customer has logged into the *TechStaff* web application.

Step	*Action*	*Expected System Response*	*Pass/ Fail*	*Comment*
1	Test Create: Click the "Create Employee" button	The system presents a list of input fields. Fields surrounded by a red box are required.		
2	Fill in required Employee fields: ID, Name, Address, Phone, Status, Years of Experience, Degree, and Expected Pay	The system enables the Save button at the bottom of the input fields		
3	Click Save.	The system presents a pop-up message "Employee #<EmployeeID>" has been saved. The list of existing contracts is displayed.		

4	*Check post-condition 1.*			
5	Test Read: Click the "Find Employee" button	The system presents a list of Employee search input fields: ID, Name, Phone, Status, Years of Experience, Degree, Expected Pay		
6	Type the desired Employee ID into the "ID" field. Click "Search"	The system presents a list of matching employees.		
7	Select the desired Employee and click the "Open" button	The system displays the Employee details. The fields are read-only.		
8	*Repeat Test Read (5) for Name, , Phone, Status, Years of Experience, Degree, and Expected Pay*			
9	Test Update: Open an Employee as described in Test Read (5) above.	The system displays the Employee details. The fields are read-only.		
10	Click the "Edit" button	The Employee fields become editable, except for the Contract ID and Name fields, which can't be changed.		
11	Enter a change for the Phone field. Click Save.	The system presents a pop-up message "Employee #<EmployeeID>" has been saved.		
12	*Check post-condition 2.*			
13	*Repeat Test Update (9) for Name, Address, Phone, Status, Years of Experience, Degree, and Expected Pay*			
14	Test Delete: Open an Employee as described in Test Read (5) above.	The system displays the Employee details. The fields are read-only.		
15	Click the "Edit" button	The Employee fields become editable, except for the Employee ID and Name fields, which can't be changed.		
16	Click the "Delete Employee" button	The system displays a pop-up message "Are you sure you want to delete Employee #<EmployeeID>? You cannot undo this action." Two buttons are presented below: "Cancel" and "Delete"		
17	Click Delete.	The system displays a message, "Employee #<EmployeeID> has been deleted."		
18	*Check post-condition 3.*			

Post-conditions:

1. The new TempEmployee is saved in the database with the entered data.
2. The new TempEmployee is saved in the database with the updated data.
3. The TempEmployee has been removed from the database.

Test case #: 4	*Test Case Name:* CRUD StaffingRequest
System: TechStaff	*Subsystem:* Employee
Designed by: Larry David	*Design Date:* 2016-11-15
Executed by: George Costanza	*Execution Date:* 2016-11-30

Short Description: Test the Creation, Reading, Updating, and Deleting of a StaffingRequest

Pre-conditions:

The Customer has received a temporary staffing demand from a Client.
The Customer has access the PSSM *TechStaff* webapp URL via web browser.

Step	*Action*	*Expected System Response*	*Pass/ Fail*	*Comment*
1	Test Create: Click the "Open Staffing Request" button	The system presents a list of input fields. Fields surrounded by a red box are required.		
2	Fill in required Staffing Request fields: ID, ContractID, Description, Experience Required, Degree Required, Pay Offered	The system enables the Save button at the bottom of the input fields		
3	Click Save.	The system presents a pop-up message "Staffing Request #<RequestID>" has been saved. The list of existing Staffing Requests is displayed.		
4	*Check post-condition 1.*			
5	Test Read: Click the "Find Staffing Request" button	The system presents a list of Staffing Request search input fields: Request ID, Status, ContractID, Experience, Degree, Pay, Reserved Employee, Placed Employee		
6	Type the desired Staffing Request ID	The system presents a list of matching		

	into the "Request ID" field. Click "Search"	Staffing Requests.		
7	Select the desired Staffing Request and click the "Open" button	The system displays the Staffing Request details. The fields are read-only.		
8	*Repeat Test Read (5) for Status, ContractID, Experience, Degree, Pay, Reserved Employee, Placed Employee*			
9	Test Update: Open a Staffing Request as described in Test Read (5) above.	The system displays the Staffing Request details. The fields are read-only.		
10	Click the "Edit" button	The Staffing Request fields become editable, except for the Request ID field, which can't be changed.		
11	Enter a change for the Pay Offer field. Click Save.	The system presents a pop-up message "Staffing Request #<RequestID>" has been saved.		
12	*Check post-condition 2.*			
13	*Repeat Test Update (9) for Status, Description, Experience Required, Degree Required, Pay Offered*			
14	Test Delete: Open a Staffing Request as described in Test Read (5) above.	The system displays the Staffing Request details. The fields are read-only.		
15	Click the "Edit" button	The Staffing Request fields become editable, except for the Employee ID and Name fields, which can't be changed.		
16	Click the "Delete Employee" button	The system displays a pop-up message "Are you sure you want to delete Staffing Request #<RequestID>? You cannot undo this action." Two buttons are presented below: "Cancel" and "Delete"		
	Click Delete.	The system displays a message, "Staffing Request #<RequestID> has been deleted."		

Post-conditions:

1. The new StaffingRequest is saved in the database with the entered data.

2. The new StaffingRequest is saved in the database with the updated data.
3. The StaffingRequest is deleted from the database.

Test case #: 5	*Test Case Name:* Reserve, place, close a Staffing Request.
System: TechStaff	*Subsystem:* Staffing Request
Designed by: Elaine Benes	*Design Date:* 2016-11-15
Executed by: Banya	*Execution Date:* 2016-11-30

Short Description: Test the candidate reservation, employee placement, and close functions of a Staffing Request.

Pre-conditions:

The Customer has received a temporary staffing demand from a Client.
The Customer has access the PSSM *TechStaff* webapp URL via web browser.
Based on a Contract search, the demand is determined to be valid.
Based on an Employee search, a suitable candidate is available.

Step	*Action*	*Expected System Response*	*Pass/ Fail*	*Comment*
1	Test Reserve: Click the "Reserve Candidate" button	The system presents a list of Employee as well as Staffing Request search input fields: Employee ID, Request ID, Employee Status, Request Status, ContractID, Employee Name, Employee Status, Expected Pay, Pay Offered, etc.		
2	Fill in desired search fields, e.g. Employee Name or Request ID	The system displays a list of items that match. For example, any employees matching the name OR any requests matching the ID are displayed.		
3	Select the desired Staffing Request and click "Candidates"	A list of available candidates is presented.		
4	Select a qualified and suitable candidate and click "Reserve."	A message is displayed: "Candidate <EmployeeID> has been reserved for Staffing Request <RequestID>." Arrangements department has been notified.		
5	*Check post-condition 1.*			

6	*Repeat Test Reserve (1, 2) as described above.*			
7	Select the desired Candidate and click "Staffing Requests"	A list of open Staffing Requests is presented.		
8	Select an appropriate Staffing Request and click "Reserve"	A message is displayed: "Candidate <EmployeeID> has been reserved for Staffing Request <RequestID>. Arrangements department has been notified."		
9	*Check post-condition 2.*			
10	Click the "Edit" button	The Staffing Request fields become editable, except for the Request ID field, which can't be changed.		
11	Enter a change for the Pay Offer field. Click Save.	The system presents a pop-up message "Staffing Request #<RequestID>" has been saved.		
12	*Check post-condition 2.*			
13	Test Place: From the *TechStaff* home screen, click "Place Employee"	A list of reserved candidates is listed.		
14	Select the desired candidate and click "Staffing Requests"	The system displays a list of open Staffing Requests.		
15	Select an appropriate Staffing Request and click the "Place" button.	A message is displayed: "Employee <EmployeeID> has been placed for Staffing Request <RequestID>, Contract <ContractID>." The Contract Manager is notified		
16	*Check post-condition 3.*			
17	From the *TechStaff* home screen, select "Search Staffing Requests"	A number of Staffing Request search fields are presented.		
18	Enter the Request ID for the placed Staffing Request from step 15 above.	The Staffing Request is displayed.		
19	Select the Staffing Request and choose "Close Request"	A message appears, "Staffing Request <RequestID> was filled and is now closed. The Client has been notified."		
	Check post-condition 4.			

Post-conditions:

1. The TempEmployee is marked as "reserved" in the database, and the associated Staffing Request now includes the EmployeeID in the ReservedEmployee field. A notification email regarding the candidate reservation was received by Arrangements.

2. The TempEmployee is marked as "reserved" in the database, and the associated Staffing Request now includes the EmployeeID in the ReservedEmployee field. A notification email regarding the candidate reservation was received by Arrangements.
3. The TempEmployee, Staffing Request, and Contract records have been updated in the database. The billing department was notified to generate a bill for the client. A copy of the Staffing Request was emailed to the Client.
4. A notification email was sent to the Client indicating that the Staffing Request was closed.

9. Change Management

A general change management recommendation plan is proposed here in order to smoothly transition PSSM from the current staffing request fulfillment solution to KMAD *TechStaff*.

9.1 Policy

Based on interviews with PSSM endusers and management, team members are eager for a better implementation of a staffing request fulfillment system. All members of PSSM's contracts, placement, and arrangements teams are affected by the poor concurrency, usability, and security of the existing spreadsheet-based solution. As such, changing policy to implement the system should be relatively straight-forward.

PSSM doesn't currently have any standard operating procedures in place around their current staffing system. Historically, new team members have learned on the job and every staff member is trained on how to coordinate with other PSSM staff to make sure data is saved to the spreadsheets reliably. New PSSM SOPs describing the use of *TechStaff* to eliminate this manual and time-consuming coordination effort should be implemented by PSSM staff after go-live to encourage adoption. These SOPs should include using *TechStaff* for:

- Managing contracts
- Maintaining existing clients
- Maintaining new clients
- Maintaining existing employee candidates
- Maintaining new employee candidates
- Predicting staffing demand with reports
- Identifying and keeping the most lucrative clients with reports

9.2 Cost/Benefit Analysis

As discussed previously, most PSSM team members are eager for a change in the tool used to fulfill staffing requests. Numerous benefits have already been identified: improved concurrency and security, elimination of unproductive coordination activities, and as a result increased productivity. PSSM Arrangements team staff are rewarded for the number of placements they are able to secure and fill, so they have a desire to work with data that is better organized, as well as saved and retrieved mored quickly.

PSSM will incur additional costs for hosting their application in the Datamart datacenter. While the costs of this investment are not insignificant, they are capitalizable as a long-term investment and over a period of 5 years PSSM finance staff have already projected a very safe return on their investment. Some additional expenses will be incurred for Training the TechStaff endusers, but a properly trained staff is necessary to use the new software effectively and realize the gains promised by the new system.

9.3 Motivation

With regard to motivation, KMAD is confident that management as well as regular staff are excited about the new system, if only to have an official reason to stop using the old one. Based on an informal census of the PSSM staff, the number of resistant and reluctant adopters of the new system is slim to none, so as long as the new system performs adequately there should no reason to ever go back to the spreadsheet-based solution.

In any case, KMAD recommends that PSSM provide with a direct way to notify KMAD about feature enhancements or fixes needed in the new system. Internal PSSM helpdesk tickets related to TechStaff can be forwarded and/or converted to the KMAD Service Desk for resolution. If PSSM staff feel like they have a voice in the direction of how their tools are built and maintained, they'll use the software actively and contribute positively to KMAD's efforts for improving it.

9.4 Training

As touched on before, making sure an effectively-trained staff is using the *TechStaff* to its fullest capabilities is in the interest of PSSM as a whole. KMAD's documentation team will provide a set of high-quality documentation pages available online for any PSSM team member to use. In addition, we are including a dedicated documentation testing phase prior to go-live as described in section 8.2.

TechStaff's efficiency is PSSM's efficiency, and the more clients and employee candidates can be matched the better for PSSM's business. KMAD's *TechStaff* was created precisely to help PSSM prosper.

10. Conclusion

KMAD TechStaff was created to meet the technical needs and requirements of Professional and Scientific Staff Management, Inc.